# Wyższa Szkoła Przedsiębiorczości i Administracji w LublinieWyższa Szkoła Przedsiębiorczości i Administracji w Lublinie

Kierunek: Informatyka

Specjalność: Projektowanie i eksploatacja systemów informatycznych

Tryb studiów: **niestacjonarny**

Poziom studiów: **I stopnia**

**Aplikacja webowa z wykorzystaniem biblioteki React**

Web application using the React library

Autor: **Damian Marek Jamróz**Nr albumu: **60192**  
Promotor: **dr inż. Sylwester Korga**

**Oświadczenie kierującego pracą.**

Oświadczam, iż niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

………………………….. …………………………..

(data) (podpis)

**Oświadczenie autora pracy**

Świadomy odpowiedzialności prawnej oświadczam, iż niniejsza praca dyplomowa została napisana przeze mnie samodzielnie nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. Nr 24, poz 83 z poźn. zm.) oraz dóbr osobistych chronionych prawem cywilnym dodatkowo nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej podmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Ponadto oświadczam, iż niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

………………………….. …………………………..

(data) (podpis)

**Spis treści**

[1. Wstęp 6](#_Toc177582276)

[1.1 Wstęp 6](#_Toc177582277)

[1.2 Cel pracy 7](#_Toc177582278)

[1.3 Zakres pracy 7](#_Toc177582279)

[1. Przegląd wykorzystanych technologii 9](#_Toc177582280)

[2.1 HTML 9](#_Toc177582281)

[2.2 JavaScript 10](#_Toc177582282)

[2.3 Typescript 11](#_Toc177582283)

[2.4 React 12](#_Toc177582284)

[2.5 React Document Object Model 14](#_Toc177582285)

[2.6 React Router DOM 15](#_Toc177582286)

[2.7 TanStack Query 16](#_Toc177582287)

[2.9 Vite 17](#_Toc177582288)

[2.10 Hypertext Transfer Protocol 18](#_Toc177582289)

[2.11 Axios 20](#_Toc177582290)

[2.12 Firebase 21](#_Toc177582291)

[2.13 Styled Components 23](#_Toc177582292)

[3. Specyfikacja aplikacji 24](#_Toc177582293)

[3.1 Użytkownicy 24](#_Toc177582294)

[3.2 Kryptowaluty 26](#_Toc177582295)

[3.3 Portfel użytkownika 28](#_Toc177582296)

[**3.3.1 Tworzenie portfolio** 29](#_Toc177582297)

[**3.3.2 Usuwanie walut z portfolio** 29](#_Toc177582298)

[**3.3.3 Dodawanie transakcji do portfolio** 29](#_Toc177582299)

[**3.3.4 Edycja transakcji w portfolio** 30](#_Toc177582300)

[**3.3.5 Usunięcie transakcji z portfolio** 30](#_Toc177582301)

[**3.3.6 Śledzenie zysków i strat** 30](#_Toc177582302)

[4. Baza danych 31](#_Toc177582303)

[5. Aplikacja serwerowa ? 32](#_Toc177582304)

[6. Aplikacja kliencka 32](#_Toc177582305)

[6.1 Struktura aplikacji 32](#_Toc177582306)

[6.2 Interfejs aplikacji 33](#_Toc177582307)

[**6.2.1 Logowanie i rejestracja** 33](#_Toc177582308)

[**6.2.2 Zakładka Portfolio** 35](#_Toc177582309)

[**6.2.2 Szczegółowy widok wybranej kryptowaluty** 38](#_Toc177582310)

[7. Podsumowanie 39](#_Toc177582311)

[Literatura 40](#_Toc177582312)

# Wstęp

W tym rodziale przedstawiony zostanie wstęp, cel oraz zakres pracy inżynierskiej.

## 1.1 Wstęp

Współczesny świat jest świadkiem rewolucji cyfrowej, która przekształca sposób, w jaki żyjemy, pracujemy i komunikujemy się. Aplikacje webowe, jako kluczowe narzędzia tej transformacji, odgrywają centralną rolę w codziennym życiu milionów ludzi. Nie są one już tylko prostymi stronami internetowymi, ale zaawansowanymi systemami, które umożliwiają realizację skomplikowanych zadań, dostęp do szerokiej gamy usług i nieograniczoną interakcję społeczną. W tym kontekście, rozwój aplikacji webowych nie jest już tylko kwestią technologiczną, ale staje się elementem kultury współczesnej, wpływającym na kształtowanie nowych modeli biznesowych, edukacyjnych i społecznych.

Wybór biblioteki React do stworzenia aplikacji webowej wynika z potrzeby odpowiedzi na te zmieniające się wymagania. React, będąc jednym z najbardziej innowacyjnych i dynamicznie rozwijających się narzędzi, umożliwia tworzenie aplikacji, które są nie tylko funkcjonalne, ale również estetycznie atrakcyjne i łatwe w obsłudze. Jego architektura oparta na komponentach pozwala na modularne budowanie aplikacji, co przekłada się na lepszą organizację kodu, łatwość w utrzymaniu i możliwość ponownego wykorzystania kodu. To sprawia, że React jest idealnym wyborem dla projektów, które wymagają szybkości, efektywności i elastyczności.

W tej pracy inżynierskiej, poprzez pryzmat biblioteki React, zostanie przedstawione, jak nowoczesne technologie webowe mogą być wykorzystane do tworzenia aplikacji, które nie tylko spełniają swoje podstawowe funkcje, ale również wzbogacają doświadczenia użytkowników i otwierają przed nimi nowe możliwości. Zostanie to osiągnięte poprzez dogłębną analizę możliwości, jakie oferuje React, oraz praktyczne zastosowanie tych możliwości w procesie tworzenia aplikacji webowej.

## 1.2 Cel pracy

Głównym celem niniejszej pracy inżynierskiej jest dogłębne zbadanie i zaprezentowanie możliwości, jakie oferuje biblioteka React w kontekście tworzenia nowoczesnych aplikacji webowych. Praca ta ma na celu nie tylko teoretyczne przedstawienie zalet Reacta, ale również praktyczne wykorzystanie tej biblioteki do zaprojektowania i stworzenia aplikacji, która będzie służyć jako narzędzie do monitorowania rynku kryptoaktywów oraz zarządzania własnym portfolio inwestycyjnym.

Założeniem Reacta, które stanowi o jego wyjątkowości, jest umożliwienie szybkiego i efektywnego tworzenia interfejsów użytkownika, które są zarówno skalowalne, jak i łatwe w utrzymaniu. Biblioteka ta, dzięki swojemu deklaratywnemu charakterowi i jednokierunkowemu przepływowi danych, umożliwia tworzenie aplikacji, które są nie tylko wydajne, ale również zapewniają doskonałe doświadczenia dla użytkowników końcowych. W kontekście rosnącego zainteresowania rynkiem kryptowalut, aplikacja webowa stworzona w ramach tej pracy będzie stanowiła praktyczny przykład wykorzystania Reacta do budowy narzędzia, które nie tylko dostarczy aktualnych informacji o cenach kryptoaktywów, ale również umożliwi użytkownikom zarządzanie ich inwestycjami w sposób intuicyjny i dostosowany do ich indywidualnych potrzeb.

Praca ta ma również na celu podkreślenie znaczenia wyboru odpowiednich narzędzi technologicznych w procesie tworzenia aplikacji webowych, które są coraz bardziej złożone i wymagające. Wybór Reacta jako głównego narzędzia do realizacji projektu nie jest przypadkowy, lecz wynika z jego zdolności do radzenia sobie z wyzwaniami współczesnego programowania webowego oraz z jego rosnącej popularności wśród programistów na całym świecie.

## 1.3 Zakres pracy

Realizacja celów określonych w pracy inżynierskiej wymaga przeprowadzenia szeregu skomplikowanych i wieloetapowych działań. Poniżej przedstawiono poszczególne etapy, które zostaną wykonane w celu osiągnięcia założonego celu:

1. **Przegląd i analiza dostępnych technologii**: Dokładne zbadanie i ocena technologii webowych dostępnych na rynku, z naciskiem na te, które są najbardziej relewantne dla tworzenia aplikacji webowych, w tym bibliotek i frameworków JavaScript.
2. **Charakteryzacja biblioteki React**: Szczegółowy opis architektury, filozofii oraz kluczowych cech Reacta, które przyczyniają się do jego popularności i efektywności w tworzeniu interfejsów użytkownika.
3. **Opracowanie wymagań funkcjonalnych i niefunkcjonalnych aplikacji**: Zdefiniowanie kluczowych funkcji, jakie ma spełniać aplikacja, oraz wymagań niefunkcjonalnych, takich jak wydajność, bezpieczeństwo i skalowalność.
4. **Projektowanie architektury bazy danych**: Stworzenie schematu bazy danych, który będzie efektywnie przechowywał dane niezbędne dla działania aplikacji.
5. **Implementacja backendu aplikacji**: Opracowanie serwera aplikacji, w tym API, które umożliwi komunikację między frontendem a bazą danych, oraz systemu logowania i autoryzacji użytkowników.
6. **Budowa modułu usługowego aplikacji**: Realizacja logiki biznesowej aplikacji, w tym algorytmów odpowiedzialnych za przetwarzanie danych i realizację funkcji aplikacji.
7. **Tworzenie interfejsu użytkownika**: Projektowanie i implementacja frontendu aplikacji z wykorzystaniem Reacta, zapewniającego intuicyjną i responsywną obsługę przez użytkowników.
8. **Testowanie manualne i automatyczne**: Przeprowadzenie testów manualnych w celu identyfikacji błędów użytkowych oraz testów automatycznych w celu zapewnienia ciągłej jakości kodu.
9. **Wstępne testy użytkowe**: Organizacja sesji testowych z udziałem potencjalnych użytkowników, aby ocenić użyteczność aplikacji i zebrać opinie, które pomogą w dalszym doskonaleniu produktu.
10. **Wdrożenie aplikacji**: Publikacja gotowej aplikacji w środowisku produkcyjnym, co umożliwi jej dostępność dla szerszego grona odbiorców.
11. **Walidacja i optymalizacja aplikacji**: Monitorowanie działania aplikacji po wdrożeniu, analiza feedbacku użytkowników oraz wprowadzanie niezbędnych poprawek i optymalizacji.

# Przegląd wykorzystanych technologii

Rozdział ten zostanie poświęcony przedstawieniu technologii, bibliotek oraz frameworków wykorzystanych do stworzenia aplikacji.

## 2.1 HTML

HTML, czyli HyperText Markup Language, jest standardowym językiem znaczników używanym do tworzenia struktur stron internetowych i aplikacji webowych. Od momentu jego stworzenia w 1991 roku przez Tima Bernersa-Lee, HTML ewoluował, dostosowując się do rosnących wymagań nowoczesnego internetu.

HTML zaczął jako prosty sposób na formatowanie i prezentowanie treści w sieci WWW (World Wide Web). Z biegiem lat, kolejne wersje HTML wprowadzały nowe znaczniki i atrybuty, które pozwalały na tworzenie coraz bardziej zaawansowanych i interaktywnych stron internetowych. HTML5, najnowsza wersja HTML, zawiera wsparcie dla multimediów, grafiki, animacji i wielu innych funkcji, które wcześniej wymagały dodatkowych wtyczek lub skomplikowanego kodu JavaScript.[1]

Podstawowe elementy HTML:

* Znaczniki: HTML używa znaczników do definiowania różnych części strony, od nagłówków i akapitów po formularze i obrazy.
* Elementy: Każdy znacznik tworzy element, który może zawierać atrybuty określające jego właściwości lub zachowanie.
* Struktura: HTML definiuje strukturę strony, umożliwiając przeglądarkom internetowym interpretowanie i wyświetlanie treści zgodnie z intencją twórcy.

HTML jest nieodłącznym elementem każdej strony internetowej i stanowi podstawę, na której budowane są bardziej złożone technologie, takie jak CSS (Cascading Style Sheets) do stylowania elementów na stronie i JavaScript do dodawania interaktywności. Współpracuje z Reactem (biblioteka JavaScript), który dynamicznie manipuluje DOM (Document Object Model) stworzonym przez HTML, aby tworzyć bogate i interaktywne widoki dla użytkownika.

## 2.2 JavaScript

JavaScript, zaprojektowany przez Brendana Eicha i wprowadzony w 1995 roku, szybko stał się nieodłącznym elementem internetu dzięki swojej zdolności do tworzenia interaktywnych stron webowych. Język ten, pierwotnie znany jako Mocha, a następnie LiveScript, ostatecznie przyjął nazwę JavaScript, stając się standardem w przeglądarkach internetowych po formalnym zatwierdzeniu standardu ECMAScript w czerwcu 1997 roku przez Ecma International.

JavaScript, przeszedł znaczącą transformację, stając się językiem, który zrewolucjonizował sposób, w jaki interaktywne strony internetowe są budowane. W 1997 roku, po przyjęciu standardu ECMAScript, JavaScript zyskał solidne fundamenty, które umożliwiły jego dalszy rozwój. Kluczowe wersje języka, takie jak ECMAScript 3 z 1999 roku, ugruntowały jego pozycję, a ECMAScript 5 z 2009 roku wprowadziło istotne ulepszenia, w tym metody dla tablic i tryb ścisły. Największym przełomem była jednak wersja ECMAScript 6 (ES6) z 2015 roku, która wprowadziła zaawansowane funkcje, takie jak klasy, moduły, let i const, funkcje strzałkowe, i promisy, co znacznie poszerzyło możliwości programistów.[2]

Jedną z kluczowych cech JavaScript jest jego dynamiczne typowanie. Oznacza to, że typy danych są określane w trakcie wykonania programu, co przynosi zarówno zalety, jak i wyzwania. Z jednej strony, dynamiczne typowanie zapewnia dużą elastyczność w tworzeniu i prototypowaniu kodu, umożliwiając programistom szybkie wprowadzanie zmian i adaptację do nowych wymagań. Z drugiej strony, może to prowadzić do trudniejszego wykrywania błędów, ponieważ problemy z typami danych mogą ujawnić się dopiero podczas wykonania programu, a nie w fazie kompilacji.

W praktyce, dynamiczne typowanie w JavaScript może prowadzić do błędów w czasie wykonania, które są trudne do zidentyfikowania i naprawienia, szczególnie w większych projektach. Brak ścisłego typowania może również wpływać na wydajność, ponieważ silnik JavaScript musi ciągle interpretować typy danych. W odpowiedzi na te wyzwania, społeczność programistyczna opracowała różne narzędzia i praktyki, aby ułatwić zarządzanie typami danych i poprawić jakość kodu. Przykładem takiego narzędzia jest TypeScript.

JavaScript jest nie tylko wszechstronny, ale również unikalny jako jedyny język, który działa natywnie w przeglądarkach internetowych, co czyni go kluczowym narzędziem dla front-end deweloperów. Umożliwia on tworzenie dynamicznych stron internetowych, manipulację DOM (aaaaaaaaaaaa), a także obsługę zdarzeń użytkownika. Dzięki Node.js, JavaScript znalazł również zastosowanie po stronie serwera, co pozwala na tworzenie kompleksowych aplikacji webowych w jednym języku. Bogaty ekosystem bibliotek i frameworków, takich jak React, Angular, Vue.js, wspiera deweloperów w tworzeniu złożonych aplikacji webowych, a narzędzia takie jak Webpack i Babel umożliwiają nowoczesne zarządzanie kodem.

JavaScript jest nieodzownym elementem współczesnych technologii webowych. Jego zastosowania obejmują nie tylko tradycyjne strony internetowe, ale także aplikacje jednostronicowe (SPA), aplikacje mobilne (za pomocą React Native), oraz aplikacje desktopowe (przy użyciu Electron). Wraz z rozwojem technologii, JavaScript znajduje również zastosowanie w dziedzinach takich jak Internet Rzeczy (IoT) oraz sztuczna inteligencja (AI), dzięki narzędziom takim jak TensorFlow.js.

JavaScript jest wspierany przez aktywną i zaangażowaną społeczność programistów. Organizowane są liczne konferencje, takie jak JSConf, które promują wymianę wiedzy i rozwój języka. Dzięki ciągłej ewolucji i wsparciu społeczności, JavaScript pozostaje jednym z najważniejszych języków programowania w świecie technologii, z perspektywą dalszego rozwoju i nowych zastosowań.

## 2.3 Typescript

W miarę rozwoju JavaScriptu i wzrostu skomplikowania aplikacji webowych, społeczność programistów zaczęła poszukiwać sposobów na ulepszenie skalowalności i zarządzania kodem. W odpowiedzi na te potrzeby, Microsoft wprowadził TypeScript w 2012 roku, nadbudowę nad JavaScriptem, która wprowadza opcjonalne statyczne typowanie i nowoczesne funkcje językowe, które są kompilowane do czystego JavaScriptu. Statyczne typowanie pozwala na wychwytywanie błędów związanych z typami danych już na etapie kompilacji, zanim kod zostanie uruchomiony, co przyczynia się do zwiększenia niezawodności aplikacji.[3]

Kluczowe cechy TypeScript:

* Statyczne typowanie: TypeScript umożliwia definiowanie typów dla zmiennych, argumentów funkcji i obiektów, co pomaga w wykrywaniu błędów przed uruchomieniem kodu.
* Interfejsy: Pozwalają na definiowanie kształtu obiektów, co jest szczególnie przydatne w dużych aplikacjach i przy pracy zespołowej.
* Klasy i dziedziczenie: TypeScript wspiera paradygmaty programowania obiektowego, takie jak klasy i dziedziczenie, co ułatwia organizację kodu.
* Dekoratory: Umożliwiają dodawanie metadanych i logiki do klas i ich członków, co może być wykorzystane w zaawansowanych wzorcach projektowych.
* Moduły: TypeScript wspiera moduły ES6, co pozwala na lepszą organizację kodu i jego ponowne wykorzystanie.
* Narzędzia: Zintegrowane narzędzia do refaktoryzacji i autouzupełniania w popularnych edytorach kodu, co znacznie przyspiesza pracę deweloperów.

TypeScript jest szczególnie ceniony w projektach, gdzie wymagana jest wysoka jakość kodu i łatwość utrzymania. Jego zastosowanie znacznie ułatwia współpracę w zespołach, gdzie różni programiści mogą pracować nad tym samym kodem. Dzięki TypeScript, programiści mogą pisać bardziej przewidywalny i łatwiejszy do zarządzania kod, co jest kluczowe w dużych aplikacjach webowych.

## 2.4 React

React, stworzony przez zespół Facebooka i po raz pierwszy zaprezentowany w 2013 roku, zrewolucjonizował sposób budowania interaktywnych interfejsów użytkownika. Jako deklaratywna biblioteka JavaScript, React umożliwia tworzenie składników (komponentów) UI (User Interface), które są zarówno efektywne, jak i elastyczne, co czyni go jednym z najbardziej wpływowych narzędzi w dzisiejszym świecie front-endu.[4]

React został stworzony, aby sprostać wyzwaniom związanym z budowaniem dużych aplikacji z dynamicznymi interfejsami użytkownika. W odpowiedzi na ograniczenia istniejących rozwiązań, inżynierowie Facebooka opracowali bibliotekę, która wprowadzała innowacyjne podejście do renderowania komponentów. Z czasem React zyskał na popularności, stając się fundamentem dla wielu aplikacji internetowych, w tym głównych produktów Facebooka, takich jak Instagram i WhatsApp.

Kluczowe cechy Reacta:

* Komponentowy model budowy aplikacji: React zachęca do tworzenia reużywalnych komponentów, które zarządzają własnym stanem, co przekłada się na modularność i łatwość w utrzymaniu kodu.
* JSX - JavaScript Syntax Extension: React wprowadził JSX, rozszerzenie składni JavaScript, które pozwala na mieszanie HTML (HyperText Markup Language) z JavaScriptem, co ułatwia tworzenie i zarządzanie interfejsem użytkownika.
* Wirtualny DOM (Document Object Model): React wykorzystuje wirtualny DOM, co pozwala na optymalizację aktualizacji interfejsu użytkownika, minimalizując interakcje z kosztownym w utrzymaniu prawdziwym DOM.
* Jednokierunkowy przepływ danych: Architektura Reacta opiera się na jednokierunkowym przepływie danych, co ułatwia zarządzanie stanem aplikacji i przewidywanie jej zachowania.

Zaawansowane możliwości Reacta:

* Hooki: Wprowadzone w wersji 16.8, hooki pozwalają na wykorzystanie stanu i innych funkcji Reacta w komponentach funkcyjnych, co otworzyło nowe możliwości w projektowaniu aplikacji.
* Context API: Umożliwia przekazywanie danych przez drzewo komponentów bez konieczności przekazywania propsów na każdym poziomie, co znacznie upraszcza zarządzanie stanem w dużych aplikacjach.

## 2.5 React Document Object Model

React DOM to część biblioteki React, która służy jako warstwa pośrednicząca między komponentami React a rzeczywistym DOM (Document Object Model) w przeglądarce. Jest to kluczowy element ekosystemu React, umożliwiający efektywne renderowanie i aktualizowanie interfejsów użytkownika w aplikacjach webowych.[4]

Rola React DOM w aplikacjach React:

* Wirtualny DOM: React używa wirtualnego DOM, który jest lekką kopią rzeczywistego DOM. Wszelkie zmiany wprowadzane w komponentach React są najpierw dokonywane na wirtualnym DOM, co pozwala na optymalizację wydajności poprzez minimalizowanie interakcji z kosztownym w utrzymaniu rzeczywistym DOM2.
* Renderowanie komponentów: React DOM odpowiada za renderowanie komponentów React na stronie internetowej, przekształcając opis interfejsu użytkownika zdefiniowany w JSX (JavaScript XML) na manipulacje DOM, które są widoczne dla użytkownika końcowego1.
* Obsługa zdarzeń: React DOM zarządza obsługą zdarzeń w aplikacji, zapewniając, że interakcje użytkowników są prawidłowo przechwytywane i przekazywane do odpowiednich komponentów.

Zalety React DOM:

* Deklaratywność: React i React DOM pozwalają na deklaratywne opisywanie interfejsów użytkownika, co ułatwia ich tworzenie, utrzymanie i skalowanie.
* Efektywność: Dzięki inteligentnemu algorytmowi porównywania, React DOM minimalizuje liczbę operacji na rzeczywistym DOM, co przekłada się na szybsze i płynniejsze działanie aplikacji.
* Uniwersalność: React DOM może być używany zarówno w aplikacjach webowych, jak i serwerowych (przy użyciu React DOM Server), co pozwala na renderowanie komponentów React zarówno po stronie klienta, jak i serwera.

## 2.6 React Router DOM

Biblioteka React Router DOM stanowi integralną część ekosystemu React i pełni rolę łącznika między komponentami React a środowiskiem przeglądarki. Jej głównym zadaniem jest umożliwienie efektywnego renderowania i aktualizowania interfejsów użytkownika w aplikacjach webowych poprzez zarządzanie trasowaniem.[5]

Kluczowe założenia React Router DOM:

Trasowanie po stronie klienta: React Router DOM umożliwia tzw. “trasowanie po stronie klienta”. W tradycyjnych witrynach internetowych przeglądarka wysyła żądanie do serwera, pobiera i ocenia pliki CSS oraz skrypty JavaScript, a następnie renderuje HTML otrzymany z serwera. W przypadku trasowania po stronie klienta, aplikacja może aktualizować adres URL po kliknięciu linku bez konieczności wysyłania nowego żądania do serwera. Zamiast tego aplikacja może natychmiast renderować nowy interfejs użytkownika i wykonywać żądania danych (np. za pomocą fetch) w celu aktualizacji strony. To pozwala na szybsze doświadczenia użytkownika, ponieważ przeglądarka nie musi pobierać całkowicie nowego dokumentu ani ponownie oceniać plików CSS i skryptów JavaScript dla kolejnej strony. Trasowanie po stronie klienta umożliwia również bardziej dynamiczne doświadczenia, takie jak animacje.

Komponenty do zarządzania trasami: React Router DOM dostarcza komponentów takich jak Route, Link, Switch i inne, które pozwalają na deklaratywne definiowanie tras i ich obsługę w aplikacji.

Zalety stosowania React Router DOM:

* Deklaratywność: React Router DOM pozwala na deklaratywne definiowanie tras w aplikacji, co ułatwia ich zrozumienie i utrzymanie.
* Wieloplatformowość: Biblioteka działa zarówno w aplikacjach webowych, jak i w aplikacjach React Native, dzięki czemu można ją używać w różnych środowiskach.
* Wsparcie społeczności: React Router DOM jest aktywnie rozwijane przez społeczność deweloperów, co zapewnia aktualność i dostosowanie do nowych standardów i praktyk w tworzeniu aplikacji webowych.

## 2.7 TanStack Query

TanStack Query, wcześniej znany jako React Query, jest jedną z najlepszych bibliotek do zarządzania stanem serwera poprzez pobieranie danych dla aplikacji webowych. W bardziej technicznych terminach, ułatwia on pobieranie, buforowanie, synchronizowanie i aktualizowanie stanu serwera w aplikacjach webowych.[6]

Podstawowe założenia TanStack Query:

* Automatyzacja procesu pobierania danych: TanStack Query automatyzuje procesy pobierania i synchronizacji danych, co pozwala deweloperom skupić się na logice biznesowej, a nie na zarządzaniu stanem serwera.
* Buforowanie i aktualizacje: Biblioteka inteligentnie zarządza buforowaniem i aktualizacjami danych, zmniejszając liczbę żądań do serwera i poprawiając wydajność aplikacji.
* Synchronizacja stanu: TanStack Query zapewnia, że stan serwera jest zawsze aktualny, synchronizując go z aplikacją w tle.

Zalety TanStack Query:

* Redukcja boilerplate’u: Używając TanStack Query, deweloperzy mogą usunąć wiele skomplikowanego i trudnego do zrozumienia kodu związane z zarządzaniem asynchronicznymi danymi.
* Łatwość utrzymania: Aplikacje stają się bardziej przewidywalne i łatwiejsze w utrzymaniu dzięki uproszczeniu procesu zarządzania danymi serwera.
* Wydajność: TanStack Query może mieć bezpośredni wpływ na odczucia użytkowników końcowych, sprawiając, że aplikacja jest szybsza i bardziej responsywna.

## 2.9 Vite

Vite jest nowoczesnym narzędziem do budowy aplikacji webowych, które zyskało uznanie dzięki swojej prędkości i wydajności. Zaprojektowany z myślą o zapewnieniu szybszego i bardziej efektywnego doświadczenia deweloperskiego dla nowoczesnych projektów webowych, Vite składa się z dwóch głównych części: serwera deweloperskiego oraz polecenia budowania.

Vite został stworzony, aby sprostać potrzebie szybszego czasu uruchamiania i gorącej wymiany modułów (Hot Module Replacement - HMR) w porównaniu do istniejących narzędzi takich jak Webpack. Jego celem jest uproszczenie procesu konfiguracji i zwiększenie prędkości rozwoju, szczególnie w dużych aplikacjach z wieloma zależnościami.[8]

Kluczowe cechy Vite:

* Szybkość i wydajność: Dzięki wykorzystaniu natywnych modułów ES w przeglądarkach, Vite oferuje błyskawiczne uruchamianie projektów i aktualizacje HMR.
* Łatwość konfiguracji: Vite jest dostarczany z sensownymi ustawieniami domyślnymi, co pozwala na szybkie rozpoczęcie pracy bez konieczności skomplikowanej konfiguracji.
* Wsparcie dla różnych frameworków: Vite może być używany z popularnymi frameworkami takimi jak Vue, biblioteką React, a także z czystym JavaScriptem, oferując szablony i wsparcie dla TypeScript, JSX i Sass.

Przepływ pracy z Vite:

1. Serwer deweloperski: Vite zapewnia serwer deweloperski z bogatymi funkcjami, takimi jak szybki HMR, który znacznie przyspiesza proces tworzenia i testowania interfejsów użytkownika.
2. Budowanie dla produkcji: Vite wykorzystuje Rollup do optymalizacji zasobów statycznych, zapewniając wydajne budowanie gotowe do wdrożenia w środowisku produkcyjnym.

Zalety stosowania Vite:

* Szybkie odświeżanie modułów: Vite zapewnia konsekwentnie szybkie aktualizacje HMR, niezależnie od rozmiaru aplikacji.
* Nowoczesne środowisko deweloperskie: Vite pozwala pominąć krok tworzenia pakietów podczas dewelopmentu, serwując przeglądarce natywne moduły ES.
* Optymalizacje wydajności: Domyślne konfiguracje Vite zapewniają wiele optymalizacji wydajnościowych w gotowych do produkcji pakietach.

Vite jest częścią rosnącego ekosystemu narzędzi front-endowych, które ułatwiają tworzenie nowoczesnych aplikacji webowych. Jego podejście do budowania projektów jest dobrze przyjęte przez społeczność deweloperów i ma duży potencjał do dalszego rozwoju i adaptacji do zmieniających się potrzeb branży webowej.

Vite, dzięki swojej prostocie i wydajności, stał się popularnym wyborem wśród deweloperów aplikacji webowych, oferując szybkie i elastyczne środowisko pracy, które jest łatwe w konfiguracji i wspiera najnowsze standardy webowe.

## 2.10 Hypertext Transfer Protocol

HTTP (Hypertext Transfer Protocol) to protokół komunikacyjny, który służy do przesyłania informacji w sieci WWW. Został stworzony przez Tim Berners-Lee i jego zespół w CERN (Europejska Organizacja Badań Jądrowych) w latach 1989-1991. HTTP jest podstawą działania internetu, umożliwiając komunikację między przeglądarkami internetowymi a serwerami.

HTTP przeszedł przez kilka wersji, każda z nich wprowadzała usprawnienia i nowe funkcje:

* HTTP/0.9 (1991): Pierwsza wersja HTTP, obsługująca tylko podstawowe żądania GET i odpowiedzi w formacie HTML.
* HTTP/1.0 (1996): Wprowadzenie wersji HTTP/1.0 przyniosło wsparcie dla dodatkowych metod (POST, HEAD) oraz wprowadziło nagłówki HTTP.
* HTTP/1.1 (1997): Ta wersja, będąca najdłużej używaną, dodała znaczące usprawnienia, takie jak utrzymywanie połączeń (persistent connections), chunked transfer encoding i inne metody (PUT, DELETE).
* HTTP/2 (2015): Wprowadzenie binarnego protokołu, który poprawił wydajność i efektywność komunikacji dzięki multiplexingowi, kompresji nagłówków i priorytetom strumieni.
* HTTP/3 (2020): Najnowsza wersja, oparta na protokole QUIC, oferuje jeszcze lepszą wydajność i bezpieczeństwo dzięki zintegrowaniu TLS i redukcji latencji.

HTTP jest protokołem bezstanowym, co oznacza, że każde żądanie klienta jest niezależne i serwer nie przechowuje żadnych informacji o wcześniejszych żądaniach. Każde żądanie zawiera wszystkie informacje potrzebne do jego obsługi. Nagłówki HTTP przekazują dodatkowe informacje o żądaniu lub odpowiedzi. Mogą zawierać metadane takie jak typ zawartości, długość, metoda żądania, informacje o kliencie i wiele innych. Są one kluczowe dla efektywnej komunikacji między klientem a serwerem.[9]

Kody statusu HTTP informują o wyniku przetwarzania żądania. Są podzielone na pięć grup:

* 1xx Informacyjne: Potwierdzają otrzymanie żądania (np. 100 Continue).
* 2xx Sukces: Oznaczają, że żądanie zostało pomyślnie przetworzone (np. 200 OK, 201 Created).
* 3xx Przekierowania: Informują o konieczności podjęcia dodatkowych działań w celu zrealizowania żądania (np. 301 Moved Permanently, 302 Found).
* 4xx Błędy klienta: Wskazują, że problem leży po stronie klienta (np. 400 Bad Request, 404 Not Found).
* 5xx Błędy serwera: Wskazują, że problem leży po stronie serwera (np. 500 Internal Server Error, 503 Service Unavailable).

Żądania HTTP składają się z trzech głównych części: pierwszej linii, nagłówków i opcjonalnie z treści (body). Pierwsza linia, zwana linią żądania, określa metodę żądania (np. GET, POST), zasób (np. ścieżkę URL), oraz wersję protokołu HTTP. Oto najważniejsze metody żądań HTTP:

* GET: Pobiera zasób z serwera.
* POST: Wysyła dane do serwera, które mają zostać przetworzone (np. w formularzach).
* PUT: Wysyła dane do serwera, zastępując istniejący zasób.
* DELETE: Usuwa zasób z serwera.
* HEAD: Pobiera nagłówki odpowiedzi bez treści.
* OPTIONS: Pobiera dostępne metody komunikacji dla zasobu.
* PATCH: Wprowadza częściowe zmiany w zasobie.

HTTP jest fundamentalnym protokołem, który umożliwia komunikację w sieci WWW. Od momentu jego wynalezienia przez Tima Berners-Lee w CERN, przeszedł liczne ewolucje, wprowadzając nowe funkcje i usprawnienia w wydajności. Jego bezstanowa natura, bogactwo nagłówków i kody statusu sprawiają, że jest niezwykle wszechstronny. Różnorodność metod żądań HTTP pozwala na elastyczne i efektywne zarządzanie komunikacją między klientami a serwerami. HTTP, będący podstawą działania internetu, pozostaje kluczową technologią w rozwijaniu nowoczesnych aplikacji webowych.

## 2.11 Axios

Axios to popularna biblioteka JavaScript, która służy do wykonywania żądań HTTP zarówno w przeglądarkach, jak i środowisku Node.js. Jest to klient HTTP oparty na Promisach, który umożliwia łatwe wysyłanie żądań asynchronicznych do serwerów i obsługę odpowiedzi. Axios jest isomorficzny, co oznacza, że może być używany zarówno po stronie klienta, jak i serwera z tą samą bazą kodu.[9]

Główne cechy Axios:

* Wsparcie dla Promisów: Axios wykorzystuje Promisy, co pozwala na eleganckie i czytelne zarządzanie asynchronicznymi operacjami.
* Interceptory żądań i odpowiedzi: Pozwalają na globalne przetwarzanie żądań i odpowiedzi przed ich ostatecznym przetworzeniem.
* Automatyczne przekształcanie danych JSON: Axios automatycznie przekształca dane JSON w żądaniach i odpowiedziach.
* Anulowanie żądań: Biblioteka oferuje mechanizmy do anulowania żądań, co jest przydatne w przypadku długotrwałych operacji.
* Ochrona przed XSRF: Axios zawiera wbudowane mechanizmy ochrony przed atakami typu Cross-Site Request Forgery (XSRF).

Zalety stosowania Axios:

* Łatwość użycia: Dzięki prostemu API, Axios jest łatwy w użyciu i konfiguracji.
* Wszechstronność: Może być używany w różnych środowiskach JavaScript, co czyni go uniwersalnym narzędziem do komunikacji z serwerem.
* Bogate API: Axios dostarcza bogaty zestaw konfiguracji, które pozwalają na dostosowanie żądań do specyficznych potrzeb aplikacji.

## 2.12 Firebase

Firebase to platforma rozwoju aplikacji stworzona przez Google, która dostarcza różne usługi i narzędzia, które pomagają w tworzeniu, rozwijaniu i skalowaniu aplikacji. Firebase jest szczególnie przydatny dla deweloperów aplikacji internetowych i mobilnych, oferując szereg funkcji, które ułatwiają rozwój i zarządzanie aplikacjami.[10]

Oto kluczowe aspekty Firebase:

* Realtime Database: Firebase oferuje bazę danych w czasie rzeczywistym, która pozwala na synchronizację danych między użytkownikami na żywo. To jest szczególnie przydatne dla aplikacji, które wymagają natychmiastowej aktualizacji danych, takich jak gry wieloosobowe, czaty na żywo i inne.
* Authentication: Firebase dostarcza łatwe w użyciu usługi uwierzytelniania, które obsługują wiele metod uwierzytelniania, takich jak uwierzytelnianie e-mail i hasło, uwierzytelnianie przez Google, Facebook, Twitter i GitHub, a także uwierzytelnianie telefonu.
* Cloud Firestore: Jest to elastyczna, skalowalna baza danych do tworzenia aplikacji mobilnych, internetowych i serwerowych. Pozwala na synchronizację danych między klientami w czasie rzeczywistym i oferuje offline support.
* Cloud Functions: Firebase oferuje możliwość tworzenia i uruchamiania funkcji serwerowych bez konieczności zarządzania infrastrukturą. Te funkcje mogą być wywoływane bezpośrednio z aplikacji lub mogą być uruchamiane w odpowiedzi na zdarzenia z innych usług Firebase.
* Hosting: Firebase oferuje szybkie i bezpieczne hosting dla aplikacji internetowych, z automatycznym SSL, CDN globalnym i funkcjami, które pomagają w szybkim wdrażaniu i serwowania treści statycznych i dynamicznych.
* Storage: Firebase oferuje skalowalne i bezpieczne przechowywanie plików dla aplikacji. Pozwala na łatwe przesyłanie i pobieranie plików, zarówno z aplikacji internetowych, jak i mobilnych.

## 2.13 Styled Components

Styled Components to innowacyjna biblioteka CSS-in-JS, która umożliwia deweloperom Reacta stylowanie aplikacji w nowy, bardziej efektywny sposób. Została stworzona z myślą o wykorzystaniu potęgi tagowanych szablonów literałów i możliwości CSS, aby zapewnić wygodne i elastyczne rozwiązanie do tworzenia komponentów ze stylami.[11]

Styled Components powstał z potrzeby uproszczenia procesu stylowania w aplikacjach React. Tradycyjne metody, takie jak zewnętrzne arkusze stylów, często prowadziły do problemów z nazewnictwem klas i utrudniały zarządzanie stylami. Styled Components rozwiązują te problemy, umożliwiając definiowanie stylów bezpośrednio w komponentach.

Kluczowe cechy Styled Components:

* Eliminacja konfliktów nazw klas: Styled Components generuje unikalne nazwy klas dla stylów, co eliminuje ryzyko konfliktów i nadpisania stylów.
* Dynamiczne stylowanie: Biblioteka pozwala na łatwe dostosowywanie stylów na podstawie propsów komponentów, co daje większą kontrolę nad wyglądem aplikacji.
* Łatwość usuwania CSS: Ponieważ każdy styl jest powiązany z konkretnym komponentem, usunięcie komponentu automatycznie usuwa wszystkie jego style.
* Automatyczne dodawanie prefiksów wendora: Styled Components zajmuje się automatycznym dodawaniem prefiksów do CSS, co zapewnia kompatybilność ze starszymi przeglądarkami.

Zalety stosowania Styled Components:

* Modularność: Stylowanie komponentów jest izolowane, co ułatwia zarządzanie kodem i jego ponowne wykorzystanie.
* Łatwość utrzymania: Deweloperzy mogą łatwiej zarządzać stylami, ponieważ są one zorganizowane w ramach komponentów, a nie rozproszone po różnych plikach.
* Kompatybilność z SSR: Styled Components wspiera renderowanie po stronie serwera (Server-Side Rendering - SSR), co jest ważne dla SEO (Search Engine Optimization) i wydajności aplikacji.

# Specyfikacja aplikacji

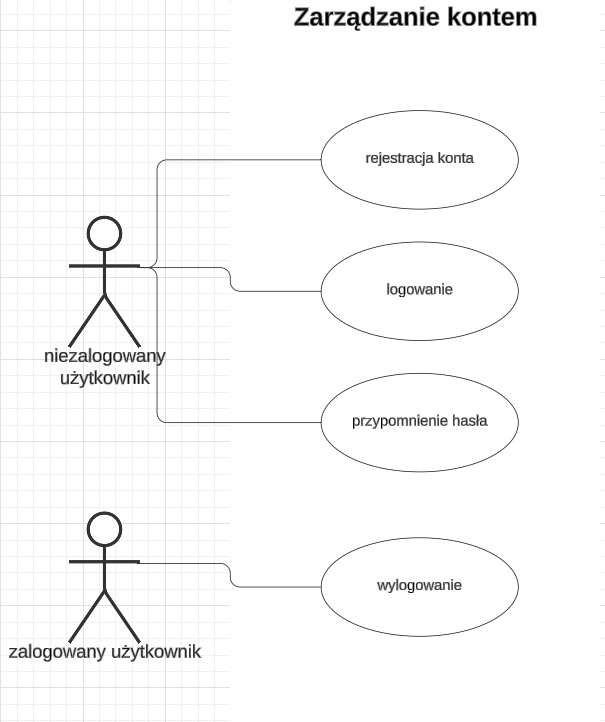
Tworzona aplikacja to internetowy tracker portfela kryptowalut. Celem aplikacji jest stworzenie platformy umożliwiającej użytkownikom przeglądanie listy kryptowalut, tworzenie własnego portfela inwestycyjnego oraz śledzenie zysków i strat w czasie rzeczywistym. Aplikacja będzie wykorzystywać bibliotekę React do budowy interfejsu użytkownika. Poniżej przedstawione są szczegółowe wymagania funkcjonalne oraz diagramy przypadków użycia, które obrazują działanie poszczególnych modułów aplikacji. Dodać info a api i be

## 3.1 Użytkownicy

Aplikacja będzie oferować różne funkcjonalności w zależności od statusu użytkownika - niezalogowanego lub zalogowanego. Poniżej przedstawiono ogólne wymagania funkcjonalne dotyczące użytkowników aplikacji:

* Rejestracja konta
* Logowanie
* Przypomnienie hasła
* Możliwość dodania oraz usunięcia wybranych walut do portfolio
* Możliwość dodania, edycji oraz usunięcia transakcji związanych z wybraną walutą z portfolio

Na podstawie tych wymagań stworzony został diagram przypadków użycia dla modułu zarządzania kontem, widoczny na rys. 3.1. Przedstawiono na nim akcje, które w aplikacji może wykonać użytkownik niezalogowany oraz zalogowany. Warto zauważyć, że dla użytkownika niezalogowanego nie będą dostępne żadne funkcjonalności poza przeglądaniem listy kryptowalut, szczegółowym widokiem konkretnej waluty, rejestracją oraz logowaniem. W kolejnych sekcjach oraz na pozostałych diagramach określenie „użytkownik” odnosi się od użytkownika zalogowanego. Rejestracja konta wymaga podania adresu e-mail oraz wpisania tego samego hasła dwa razy. Logowanie odbywa się poprzez podanie adresu e-mail oraz hasła. Użytkownik zalogowany uzyskuje dostęp do nowej zakładki ‘Portfolio’ w której ma możliwość stworzenia własnego portfela kryptowalut oraz śledzenie zysków i strat.



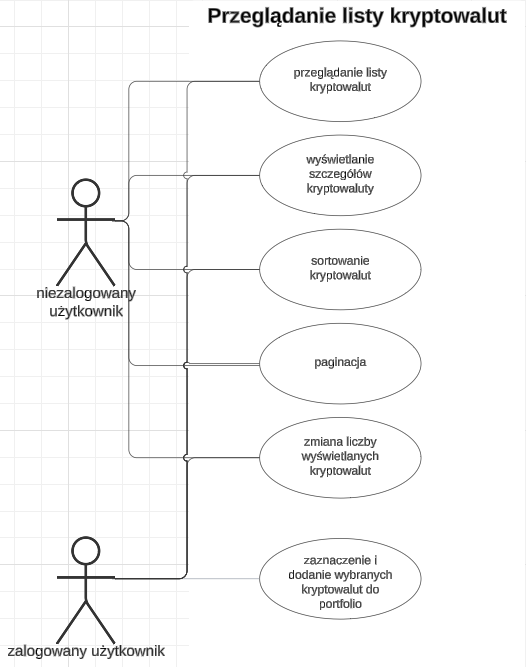
Rysunek 3.1: Diagram przypadków użycia modułu zarządzania kontem użytkownika

## 3.2 Kryptowaluty

Każdy użytkownik, niezależnie od statusu logowania, może przeglądać listę kryptowalut, która zawiera kluczowe informacje, takie jak nazwa, symbol, aktualna cena i kapitalizacja rynkowa. Dodatkowo, można przejść do szczegółów wybranej waluty, aby zobaczyć więcej danych. Użytkownik ma także możliwość sortowania kryptowalut według różnych kryteriów (np. cena, kapitalizacja), korzystania z paginacji oraz wyboru liczby wyświetlanych walut na stronie (np. 10, 20, 50). Te funkcje są stworzone z myślą o łatwiejszej nawigacji i szybkim znajdowaniu interesujących kryptowalut.

Dla użytkowników zalogowanych dostępne są te same opcje, z dodatkową możliwością dodawania wybranych walut do swojego portfolio.

Aplikacja korzysta z CoinGecko API (<https://docs.coingecko.com/v3.0.1/reference/introduction>) w celu pobierania danych o kryptowalutach, takich jak ich aktualna cena, kapitalizacja rynkowa, wolumen handlu, oraz inne szczegóły. Dane te są następnie prezentowane na stronie głównej oraz w szczegółowym widoku wybranej kryptowaluty.



Rysunek 3.2: Diagram przypadków użycia modułu przeglądania listy kryptowalut

## 3.3 Portfel użytkownika

Zalogowany użytkownik ma możliwość tworzenia i zarządzania swoim portfelem kryptowalut w zakładce ‘Portfolio’. Poniżej przedstawiono wymagania funkcjonalne dotyczące zarządzania portfelem. Do zarządzania danymi użytkowników, w tym ich portfolio i transakcjami, wykorzystano Firebase Database. Baza danych przechowuje informacje o portfelach, takich jak dodane kryptowaluty oraz szczegóły transakcji. Firebase jest również używany do obsługi systemu logowania i rejestracji użytkowników.

### **3.3.1 Tworzenie portfolio**

Użytkownik może utworzyć nowe portfolio, dodając tam wybrane waluty z listy dostępnej na stronie głównej. Wystarczy zaznaczyć interesujące waluty i kliknąć przycisk "Dodaj do portfolio". Po udanej operacji użytkownik otrzyma powiadomienie, a wybrane waluty pojawią się w zakładce "Portfolio". Portfolio może zawierać dowolną liczbę kryptowalut.

### **3.3.2 Usuwanie walut z portfolio**

W zakładce "Portfolio" użytkownik może zaznaczyć wybrane waluty i usunąć je z portfela. Można również usunąć wszystkie waluty naraz. Przed wykonaniem tej operacji pojawi się okno dialogowe z prośbą o potwierdzenie. Po potwierdzeniu, użytkownik otrzyma powiadomienie o sukcesie operacji lub o błędzie, jeśli coś poszło nie tak.

### **3.3.3 Dodawanie transakcji do portfolio**

Aby dodać transakcję do wybranej waluty, użytkownik klika przycisk z symbolem ołówka obok waluty. Pojawia się okno dialogowe, w którym należy podać ilość jednostek oraz cenę zakupu. Użytkownik może także kliknąć przycisk "CENA RYNKOWA", który automatycznie uzupełni pole aktualną ceną rynkową. Po zapisaniu transakcji, użytkownik otrzyma powiadomienie o powodzeniu operacji lub o błędzie w przypadku problemów z zapisem.

### **3.3.4 Edycja transakcji w portfolio**

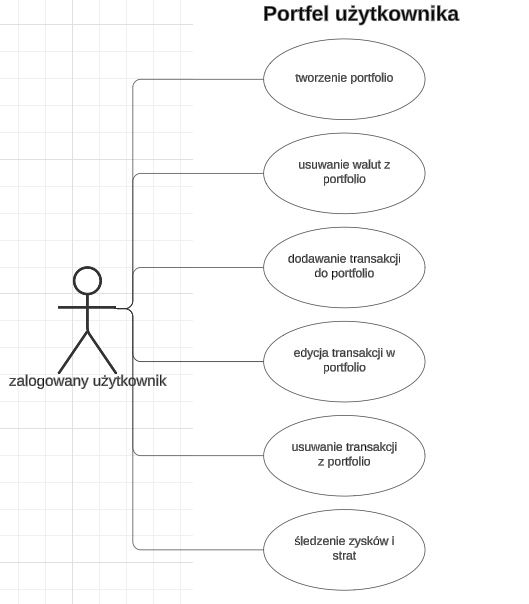
Aby edytować transakcję, użytkownik ponownie klika przycisk z symbolem ołówka przy walucie, której dotyczy transakcja. W oknie dialogowym pojawią się wcześniej wprowadzone dane, które można zmodyfikować. Po zapisaniu zmian, użytkownik zostanie poinformowany o sukcesie lub ewentualnym błędzie.

### **3.3.5 Usunięcie transakcji z portfolio**

Użytkownik może usunąć wybraną transakcję, klikając przycisk z symbolem kosza obok niej. Pojawi się okno dialogowe z prośbą o potwierdzenie. Po usunięciu transakcji, aplikacja wyświetli powiadomienie o powodzeniu operacji.

### **3.3.6 Śledzenie zysków i strat**

Aplikacja umożliwia użytkownikowi śledzenie zysków i strat na podstawie zmian cen kryptowalut. System wyświetla sumę wydatków oraz aktualny zysk lub stratę. Pokazuje również szczegóły dotyczące każdej waluty – jej ilość oraz wartość, zarówno w formie tabeli, jak i wykresu. Aplikacja automatycznie oblicza wartość portfela na podstawie aktualnych cen rynkowych.



Rysunek 3.3: Diagram przypadków użycia modułu zarządzania portfelem

# Baza danych

W aplikacji wykorzystano dwa główne źródła danych: CoinGecko API oraz Firebase Realtime Database. Choć API CoinGecko dostarcza informacji o rynku kryptowalut, takich jak kursy walut, kapitalizacja rynkowa czy wolumeny handlu, to dane te nie są przechowywane w bazie danych aplikacji. Są one jedynie wyświetlane użytkownikom w czasie rzeczywistym. Z kolei dane związane z użytkownikami, ich portfelami oraz transakcjami są zarządzane za pomocą Firebase. --- zmienić wstęp na bardziej wstępowy

## 4.1 CoinGecko API

CoinGecko API zapewnia dostęp do aktualnych danych rynkowych dotyczących kryptowalut, takich jak ceny, wolumen handlu, kapitalizacja rynkowa czy dostępna podaż. API to jest wykorzystywane do dynamicznego wyświetlania danych na stronie głównej aplikacji oraz w szczegółowym widoku kryptowaluty (rozdział 6.2.2). Dzięki temu aplikacja oferuje użytkownikom aktualne informacje o kryptowalutach bez potrzeby przechowywania tych danych lokalnie.

## 4.2 Firebase Realtime Database

Firebase Realtime Database jest głównym rozwiązaniem przechowywania danych w aplikacji. Jest to baza danych w czasie rzeczywistym, co oznacza, że wszelkie zmiany danych są natychmiast widoczne na wszystkich podłączonych urządzeniach użytkowników.

### **4.2.1 Logowanie i rejestracja**

Firebase Authentication obsługuje procesy logowania i rejestracji, pozwalając użytkownikom na bezpieczne tworzenie kont, logowanie się, a także resetowanie zapomnianego hasła. Dane użytkowników są bezpiecznie przechowywane w Firebase, co zapewnia wygodny dostęp do kont z dowolnego urządzenia.

### **4.2.2 Portfele użytkowników**

Dane dotyczące portfela, czyli listy wybranych kryptowalut, są przechowywane w Firebase. Każdy użytkownik ma swój indywidualny portfel, który zawiera informacje o wybranych kryptowalutach oraz szczegóły transakcji.

### **4.2.3 Transakcje**

Każda transakcja wykonana przez użytkownika jest zapisywana w bazie danych Firebase. Transakcje obejmują takie informacje jak liczba zakupionych jednostek kryptowaluty, cena zakupu oraz data transakcji. Firebase pozwala na edycję, usuwanie oraz dodawanie nowych transakcji, co jest natychmiastowo odzwierciedlane w interfejsie użytkownika.

## 4.3 Struktura danych w Firebase

Dane w Firebase są przechowywane w formacie JSON i mają strukturę drzewa. Każdy użytkownik ma swój unikalny identyfikator (UID), który jest kluczem głównym dla przechowywania danych w bazie.

# Aplikacja serwerowa ?

Tworzona aplikacja to do bogatej bazy wiedzy.

# Aplikacja kliencka

Rozdział ten poświęcony jest frontendowej części aplikacji. Najpierw krótko przedstawiono strukturę aplikacji, a następnie omówiono istotne elementy i funkcjonalności interfejsu aplikacji.

## 6.1 Struktura aplikacji

Aplikacja kliencka została stworzona z wykorzystaniem biblioteki React, która pozwala na logiczny podział interfejsu użytkownika na komponenty. Komponenty te zawierają zarówno kod JSX (połączenie JavaScript i HTML), jak i logikę zarządzania stanem aplikacji. Dzięki temu struktura aplikacji jest przejrzysta i modularna, co ułatwia jej rozwój i utrzymanie.

Do komunikacji z backendem wykorzystano bibliotekę Axios, która ułatwia wykonywanie zapytań HTTP oraz zarządzanie danymi asynchronicznymi. Zapytania do API są zorganizowane w osobnych funkcjach pomocniczych, co umożliwia łatwe zarządzanie danymi i ich ponowne wykorzystywanie w różnych komponentach. W aplikacji za implementację routingu odpowiada React Router, który pozwala na dynamiczne nawigowanie między różnymi widokami.

React opiera się na koncepcji "state" i "props", co pozwala efektywnie zarządzać stanem aplikacji i przekazywać dane między komponentami. Dzięki zastosowaniu narzędzia React Query, proces pobierania, buforowania i aktualizowania danych jest zoptymalizowany, co pozwala na efektywne zarządzanie stanem aplikacji w czasie rzeczywistym. Wszystko to działa w połączeniu z narzędziem Vite, które wspiera szybkie kompilowanie i budowanie aplikacji.

## 6.2 Interfejs aplikacji

Ta sekcja zawiera przegląd interfejsu i funkcjonalności aplikacji klienckiej.

### **6.2.1 Logowanie i rejestracja**

Na rysunku XX pokazano panel logowania. Aby zalogować się na swoje konto, użytkownik musi wprowadzić zarówno adres e-mail, jak i hasło. Oba pola są obowiązkowe. Po wypełnieniu formularza należy nacisnąć przycisk „Zaloguj się”, aby kontynuować. Jeśli podane dane są nieprawidłowe, takie jak błędne hasło lub nieznany adres e-mail, po weryfikacji przez backend, pojawi się odpowiedni komunikat informujący o problemie.

W przypadku potrzeby utworzenia nowego konta, wystarczy kliknąć link umieszczony poniżej przycisku logowania. Dodatkowo, jeśli użytkownik zapomniał hasła, może skorzystać z opcji „Zapomniałeś hasła?”, która przekieruje go do formularza umożliwiającego odzyskanie dostępu do konta widocznego na rysunku XXX. Użytkownik musi wprowadzić adres e-mail przypisany do konta, na który zostanie wysłana wiadomość z linkiem do resetowania hasła. Po kliknięciu w ten link, użytkownik zostanie przeniesiony do strony widocznej na rysunku XX, na której będzie musiał wprowadzić nowe hasło. Po pomyślnym zresetowaniu hasła, użytkownik będzie mógł zalogować się do aplikacji, używając nowo ustalonego hasła.

Rysunek XX: Panel z możlwością zresetowania hasła

Rysunek XX: Panel w którym użytkownik może ustawić nowe hasło

Na rysunku XX widoczny jest panel rejestracji. Formularz składa się z trzech pól tekstowych: adresu e-mail oraz dwóch pól do wprowadzenia hasła. Hasło musi mieć co najmniej sześć znaków, natomiast e-mail musi zawierać znak @. Jeśli te kryteria nie zostaną spełnione, pojawi się odpowiedni komunikat. Aby zarejestrować nowe konto, użytkownik powinien nacisnąć przycisk „Zarejestruj się”. W sytuacji, gdy wprowadzony adres e-mail jest już zarejestrowany, wyświetli się informacja o błędzie. W przeciwnym razie konto zostanie pomyślnie utworzone, a użytkownik automatycznie zostanie przekierowany na główny ekran aplikacji.

Rysunek XX: Panel logowania

Rysunek XX: Panel rejestracji

Na rysunku XXX przedstawiono główną stronę aplikacji, której wygląd różni się w zależności od tego, czy użytkownik jest zalogowany, czy nie. W lewym górnym rogu zawsze widoczne jest logo aplikacji, a obok niego znajduje się panel kanałów. Dla użytkowników zalogowanych pojawia się jednak dodatkowa zakładka „Portfolio”, umożliwiająca dostęp do indywidualnego portfela kryptowalut.

Po prawej stronie, na górze ekranu, umieszczona jest nawigacja, pod którą widoczne są podstawowe dane rynkowe, takie jak kapitalizacja i ceny. Te same dane są również prezentowane po lewej stronie centralnej części ekranu, w nieco innej formie, bardziej szczegółowej. Z kolei na środku po prawej stronie znajduje się lista najpopularniejszych kryptowalut przedstawiona w formie tzw. karuzeli.

Pod tą sekcją wyświetlana jest tabela z pełną listą kryptowalut. Użytkownicy zalogowani mają możliwość zaznaczania dowolnej ilości walut w tej tabeli i dodawania ich do swojego portfela. Natomiast użytkownicy niezalogowani mogą jedynie przeglądać tę listę.

Na samym dole strony znajduje się paginacja oraz opcja wyboru liczby wyświetlanych kryptowalut, a poniżej – stopka z opisem strony.

Rysunek XX: Strona główna

### **6.2.2 Zakładka Portfolio**

Zakładka Portfolio, widoczna na rysunku XXX, to kluczowy element aplikacji, w którym użytkownik może zarządzać swoimi inwestycjami w kryptowaluty. Jak widać na ilustracji, po lewej stronie na górze ekranu znajduje się logo aplikacji, a po prawej stronie umieszczono nawigację, co ułatwia dostęp do innych sekcji. Bezpośrednio pod nawigacją, podobnie jak na stronie głównej, znajdują się podstawowe dane rynkowe, takie jak aktualne ceny najpopularniejszych kryptowalut oraz ich zmiany. Te informacje wyświetlane są w przystępnej formie, co pomaga użytkownikowi śledzić rynek na bieżąco.

W centralnej części ekranu, poniżej danych rynkowych, znajduje się krótka instrukcja, która informuje użytkownika, że w tej zakładce może dodawać kryptowaluty do swojego portfela oraz rejestrować transakcje. Jest to istotny punkt startowy dla użytkownika, który jeszcze nie wprowadził żadnych danych, a rysunek XXX dokładnie ilustruje wygląd tego widoku.

Rysunek XX: Zakładka portfolio bez danych użytkownika

Poniżej nagłówka „Twoje portfolio”, po lewej stronie, prezentowane są sumaryczne dane dotyczące inwestycji: „Suma wydatków: 0.00 USD” oraz „Całkowity zysk/strata: 0.00 USD”. Te wartości będą aktualizowane w czasie rzeczywistym po dodaniu transakcji. Po prawej stronie tego sekcji znajduje się wykres kołowy, który można zobaczyć dopiero po wprowadzeniu danych — na przykład, gdy użytkownik doda jakąkolwiek kryptowalutę do portfela. Rysunek XXX przedstawia przykładowy widok portfela z aktywnym wykresem.

Rysunek XX: Zakładka portfolio z danymi użytkownika

Poniżej znajdują się dwa kluczowe przyciski: „Usuń zaznaczone waluty” oraz „Usuń wszystkie waluty”. Przycisk „Usuń zaznaczone waluty” pozostaje nieaktywny, dopóki użytkownik nie zaznaczy jednej lub więcej walut w swoim portfelu, co zobrazowano na rysunku XXX. Oba przyciski służą do usuwania walut z portfela — jeśli użytkownik zdecyduje się na usunięcie waluty, zostaną również usunięte wszystkie transakcje z nią związane. Po kliknięciu w jeden z dostępnych przycisków użytkownik zobaczy okno dialogowe z pytanie o potwierdzenie tej akcji. Widać to na rysunku XXX.

Rysunek XX: Zakładka portfolio z aktywnymi przyciskami

Rysunek XX: Okno dialogowe przy próbie usunięcia jakiejs kryptowaluty

Pod tymi przyciskami znajduje się tabela z wybranymi przez użytkownika kryptowalutami. Jak widać na rysunku XXX, każdy wiersz tabeli reprezentuje jedną kryptowalutę, a użytkownik może ją zaznaczyć, aby ją usunąć, lub kliknąć na jej nazwę, aby przejść do szczegółowych informacji, podobnych do tych dostępnych na stronie głównej. Dodatkowo, tabela umożliwia sortowanie kryptowalut według różnych kryteriów, takich jak nazwa, wartość, czy zmiana ceny, co działa w identyczny sposób jak sortowanie dostępne na głównej stronie aplikacji, co zilustrowano na rysunku XXX.

Rysunek XX: Tabela z wybranymi kryptowalutami

Rysunek XX: Tabela z wybranymi kryptowalutami oraz przykładem sortowania

Przy każdej kryptowalucie znajduje się również opcja dodania transakcji poprzez kliknięcie przycisku z symbolem ołówka, co pokazuje rysunek XXX. Po kliknięciu pojawia się okno dialogowe, w którym użytkownik może wpisać ilość oraz cenę zakupionych jednostek kryptowaluty. Dodatkowo, tuż pod polem do wprowadzania ceny znajduje się przycisk „Cena rynkowa”, który automatycznie uzupełnia cenę o aktualną wartość rynkową danej kryptowaluty. Na dole okna znajduje się przycisk „Dodaj transakcję”, który aktywuje się, gdy oba pola są poprawnie wypełnione — czyli wartość ilości oraz ceny są wyższe niż zero. Rysunek XXX przedstawia przykład takiego formularza.

Rysunek XX: Przycisk z symbolem ołówka

Rysunek XX: Okno dialogowe pozwalające na dodanie lub edycje transakcji

Dodanie transakcji wpływa na wygląd całej zakładki. Zaktualizowane zostają wartości wydatków i zysku/straty oraz ilość posiadanych kryptowalut, co zobrazowano na wykresie kołowym. W tabeli widoczne są szczegóły każdej kryptowaluty, takie jak ilość zakupionych jednostek oraz ich aktualna wartość rynkowa.

Oprócz dodawania, użytkownik ma możliwość edytowania transakcji. Ponowne kliknięcie przycisku z symbolem ołówka otworzy okno z wcześniej wprowadzonymi danymi, co umożliwia ich modyfikację. Przedstawia to rysunek XXX. Dla każdej kryptowaluty można także usunąć transakcje, klikając w przycisk z symbolem kosza, co zobrazowano na rysunku XXX. Pojawia się wtedy okno dialogowe z pytaniem o potwierdzenie tej akcji.

Rysunek XX: Okno dialogowe podczas edycji

Rysunek XX: Okno dialogowe z potwierdzeniem usunięcia wybranej transakcji

Na samym dole zakładki, podobnie jak na stronie głównej, znajduje się stopka zawierająca opis strony oraz dostępne zakładki.

### **6.2.2 Szczegółowy widok wybranej kryptowaluty**

Zakładka szczegółowego widoku kryptowaluty, jak widać na rysunku XXX, jest miejscem, gdzie użytkownik może uzyskać pełne informacje na temat wybranej przez siebie waluty. Po kliknięciu w nazwę kryptowaluty w tabeli (np. Bitcoin), użytkownik zostaje przekierowany do nowego widoku, który przedstawia szczegółowe dane dotyczące wybranej kryptowaluty.

Widok ten podzielony jest na dwie główne sekcje. Po lewej stronie znajdują się kluczowe informacje o kryptowalucie, jak pokazano na rysunku XXX. Na przykładzie Bitcoina możemy zobaczyć takie dane jak:

* Nazwa kryptowaluty i jej ranking (np. „Bitcoin Kurs BTC #1”),
* Aktualna cena (np. „59595 USD”),
* Kapitalizacja rynkowa (np. „1,177,353,803,147 USD”),
* Wolumen handlu w ciągu ostatnich 24 godzin (np. „32,979,223,741 USD”),
* Ilość zasobów w obiegu (np. „19,755,815”),
* Maksymalna ilość zasobów, jakie mogą być dostępne (np. „21,000,000”).

Te informacje pomagają użytkownikowi zrozumieć aktualny stan danej kryptowaluty oraz jej znaczenie na rynku. Jak pokazano na rysunku XXX, dane są wyświetlane w czytelny i zwięzły sposób, co ułatwia szybki przegląd najważniejszych aspektów związanych z wybraną walutą.

Rysunek XXX: Szczegółowy widok kryptowaluty - dane

Po prawej stronie, widoczny jest interaktywny wykres przedstawiający zmiany wartości danej kryptowaluty w ciągu ostatnich 24 godzin, co zostało zilustrowane na rysunku XXX. Wykres ten pozwala użytkownikowi śledzić zmiany cen w czasie rzeczywistym, co może być kluczowe dla podejmowania decyzji inwestycyjnych.

Rysunek XXX: Szczegółowy widok kryptowaluty - wykres wartości

# Podsumowanie

Celem pracy było zaprezentowanie możliwości frameworka React do tworzenia aplikacji internetowych oraz zaprojektowanie i wykonanie aplikacji z nim.

Stworzona aplikacja to strona umożliwiający użytkownikom śledzenie wartości rynkowej walut w czasie rzeczywistym oraz tworzenie własnego portfolio. Cechą aplikacji jest prostota struktury oraz możliwość jej rozbudowy. Korzystanie z gotowych rozwiązań pozwala na szybkie dodanie nowych funkcji. Niezwykle ważną kwestią jest również przejrzystość struktury i sposobu działania. Pozwala to zaprezentować sprawnie działającą i użyteczną aplikację wykorzystującą architekturę REST oraz Websocket.

Cel pracy został osiągnięty. Stworzona aplikacja spełnia wymagania postawione w specyfikacji.

Autor za własny wkład pracy uważa:

- przygotowanie opisu oraz przeglądu technologii wykorzystanych do stworzenia aplikacji,

- określenie wymagań tworzonej aplikacji,

- implementacja zaprojektowanej aplikacji,

- przygotowanie dokumentacji opisującej strukturę oraz interfejs stworzonej aplikacji.

# Literatura

[1] HTML: HyperText Markup Language.

https://developer.mozilla.org/en-US/docs/Web/HTML. Dostęp data.

[2] JavaScript.

https://developer.mozilla.org/en-US/docs/Web/JavaScript. Dostęp data.

[3] TypeScript for JavaScript Programmers.

https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html. Dostęp data.

[4] React.

<https://react.dev/reference/react>. Dostęp data.

[5] React Router DOM.

https://reactrouter.com/en/main/start/overview. Dostęp data.

[6] TanStack Query (React Query).

https://tanstack.com/query/v4/docs/framework/react/overview. Dostęp data.

[8] Vite.

https://vitejs.dev/guide/. Dostęp data.

[9] HTTP.

https://developer.mozilla.org/en-US/docs/Web/HTTP

[10] Axios.

https://axios-http.com/docs/intro. Dostęp data.

[11] Firebase.

https://firebase.google.com/docs?hl=pl. Dostęp data.

[12] Styled Components.

https://styled-components.com/docs/basics. Dostęp data.

[13] Chart.js.

https://www.chartjs.org/docs/latest/. Dostęp data.

[14] Slick Carousel.

https://kenwheeler.github.io/slick/ Dostęp data.